GBDT

汇报人：扣扣

大纲

- 集成学习
- Bagging
- Boosting
- 提升树
- 梯度提升树GBDT
- 扩展：Xgboost

集成学习

- 人多力量大
- 3 个臭皮匠顶个诸葛亮
- 集成学习并不创造出新的算法，而是把已有的算法进行结合，从而得到更好的效果。
- 集成学习是一种“思路”，并不是某种具体的方法或者算法。
- 随机森林、多头机制、dropout、采样...

京东智联云 & 贪心科技 GREEDY TECHNOLOGY
NeuHub AI Links All

Bagging

- 民主模式：每轮从原始样本集中使用 Bootstraping 的方法抽取 n 个训练样本，共进行 k 轮抽取，得到 k 个训练集，每次使用一个训练集得到一个模型，k 个训练集共得到 k 个模型。最终的结果使用民主投票的方式得到。
- bagging + 决策树 = 随机森林

AI Links All

Boosting

- 精英模式：在每一轮训练中，改变训练数据的权值或概率分布，通过提高那些在前一轮被分错样例的权值，减小前一轮分对样例的权值，来使得分类器对误分的数据有较好的效果（不断学习、不断提升的过程）。
- 通过加法模型将基础模型进行加权组合。

提升树（Boosting Tree）

- 提升树模型是一种Boosting方法。
- 提升树模型采用加法模型（基函数的线性组合）与前向分步算法，同时基函数采用决策树算法，提升树模型可以看作是决策树的加法模型。
- 树+集成（互补）

提升树（Boosting Tree）

- 初始化$f_0(x)$
- 对m = 1, 2, ..., M
  - 计算残差：$r_{mi} = y_i - f_{(m-1)}(x), i = 1, 2, \ldots, N$
  - 拟合残差学习一个回归树，即$h_m(x)$
  - 更新$f_m(x) = f_{m-1} + h_m(x)$
- 得到回归提升树：
  - $f_M(x) = \sum_{m=1}^{M} h_m(x)$

NeuHub AI Links All

梯度提升树（GBDT）

- GBDT又称为：GBT（Gradient Boosting Tree），GTB（Gradient Tree Boosting），GBRT（Gradient Boosting Regression Tree），MART(Multiple Additive Regression Tree)。
- GBDT在工业界有广泛应用，可认为是机器学习算法中的top3。

NeuHub
AI Links All

梯度提升树（GBDT）

- 与提升树最大的不同：使用负梯度而非残差，作为每一步拟合的对象
- 初始化$f_0(x)$
- 对m = 1, 2, ..., M
  - 计算负梯度：$r_{mi} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}], i = 1, 2, \ldots, N, f(x) = f_{m-1}(x)$
  - 拟合负梯度学习一个回归树，即$h_m(x)$
  - 更新$f_m(x) = f_{m-1} + h_m(x)$
- 得到回归提升树：
- $f_M(x) = \sum_{m=1}^{M} h_m(x)$

NeuHub AI Links All

梯度提升树（GBDT）

- 当损失函数为mse时：
  - 计算负梯度：$r_{mi} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}], i = 1, 2, \ldots, N, f(x) = f_{m-1}(x)$
  - $L(y_i, f(x_i)) = 0.5 * (y_i - f(x_i))^2$
  - $r_{mi} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}] = y_i - f(x_i)$
  - 负梯度即残差！
  - 回归提升树可以看作GBDT在损失函数为MSE时的特例

NeuHub AI Links All

梯度提升树（GBDT）

- 特征为年龄、体重，身高为标签值。共有5条数据，前四条为训练样本，后一条为要预测的样本。

编号	年龄(岁)	体重（kg）	身高(m)(标签值)
0	5	20	1.1
1	7	30	1.3
2	21	70	1.7
3	30	60	1.8
4(要预测的)	25	65	?

梯度提升树（GBDT）

- 参数设置：
  - 学习率：learning_rate=0.1
  - 迭代次数：n_trees=5
  - 树的深度：max_depth=3

NeuHub
AI Links All

梯度提升树（GBDT）

- 初始化弱学习器（只有一个节点）
- $f_0(x) = arg\,min_c \sum_{i=1}^{N} L(y_i, c)$
- *损失函数为平方损失，是凸函数，直接求导，倒数等于零，得到c:*
- $\sum_{i=1}^{N} \frac{\partial L(y_i,c)}{\partial c} = \sum_{i=1}^{N} \frac{\partial (0.5*(y_i - c)^2)}{\partial c} = \sum_{i=1}^{N} c - y_i = 0$
- *c=(1.1+1.3+1.7+1.8)/4=1.475*
- $f_0(x) = arg\,min_c \sum_{i=1}^{N} L(y_i, c) = c = 1.475$

NeuHub AI Links All

梯度提升树（GBDT）

- 对迭代轮数m=1，2,...,M:

由于设置了迭代次数：n_trees=5，这里的M=5,计算负梯度，当损失函数为平方损失时，负梯度就是残差，即y与上一轮得到的学习器fm−1的差值。

编号	真实值	$f_0(x)$	残差
0	1.1	1.475	-0.375
1	1.3	1.475	-0.175
2	1.7	1.475	0.225
3	1.8	1.475	0.325

梯度提升树（GBDT）

- 寻找回归树的最佳划分节点：遍历每个特征的每个可能取值。从年龄特征的5开始，到体重特征的70，分别计算分裂后两组数据的平方损失。

编号	年龄(岁)	体重（kg）	标签值
0	5	20	-0.375
1	7	30	-0.175
2	21	70	0.225
3	30	60	0.325

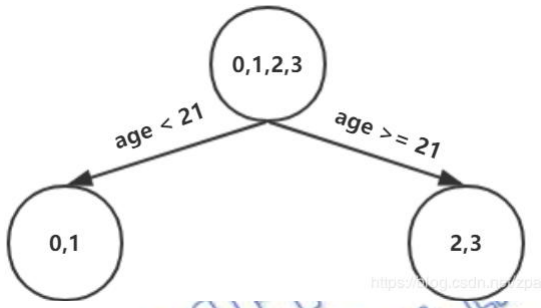
梯度提升树（GBDT）

划分点	小于划分点的样本	大于等于划分点的样本	SE_l	SE_r	SE_{sum}
年龄5	/	0，1，2，3	0	0.327	**0.327**
年龄7	0	1，2，3	0	0.140	**0.140**
年龄21	0，1	2，3	0.020	0.005	**0.025**
年龄30	0，1，2	3	0.187	0	**0.187**
体重20	/	0，1，2，3	0	0.327	**0.327**
体重30	0	1，2，3	0	0.140	**0.140**
体重60	0，1	2，3	0.020	0.005	**0.025**
体重70	0，1，3	2	0.260	0	**0.260**

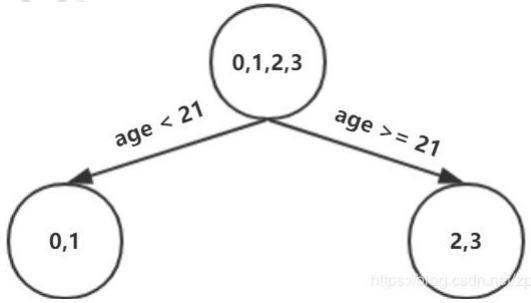
梯度提升树（GBDT）

划分点	小于划分点的样本	大于等于划分点的样本	SE_l	SE_r	SE_{sum}
年龄5	/	0，1，2，3	0	0.327	**0.327**
年龄7	0	1，2，3	0	0.140	**0.140**
年龄21	0，1	2，3	0.020	0.005	**0.025**
年龄30	0，1，2	3	0.187	0	**0.187**
体重20	/	0，1，2，3	0	0.327	**0.327**
体重30	0	1，2，3	0	0.140	**0.140**
体重60	0，1	2，3	0.020	0.005	**0.025**
体重70	0，1，3	2	0.260	0	**0.260**

总平方损失最小为0.025有两个划分点：年龄21和体重60，随机选一个作为划分点。



梯度提升树（GBDT）

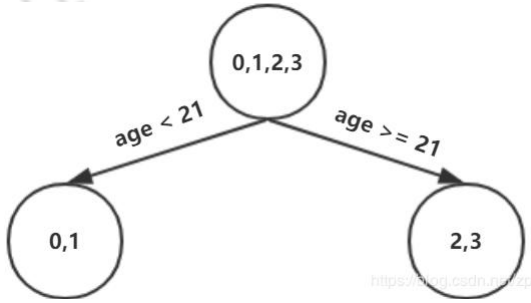


划分点	小于划分点的样本	大于等于划分点的样本	SE_l	SE_r	SE_{sum}
年龄5	/	0，1	0	0.020	**0.020**
年龄7	0	1	0	0	**0**
体重20	/	0，1	0	0.020	**0.020**
体重30	0	1	0	0	**0**

对于左节点，只含有0,1两个样本，根据表选择年龄7划分。

NeuHub AI Links All

梯度提升树（GBDT）

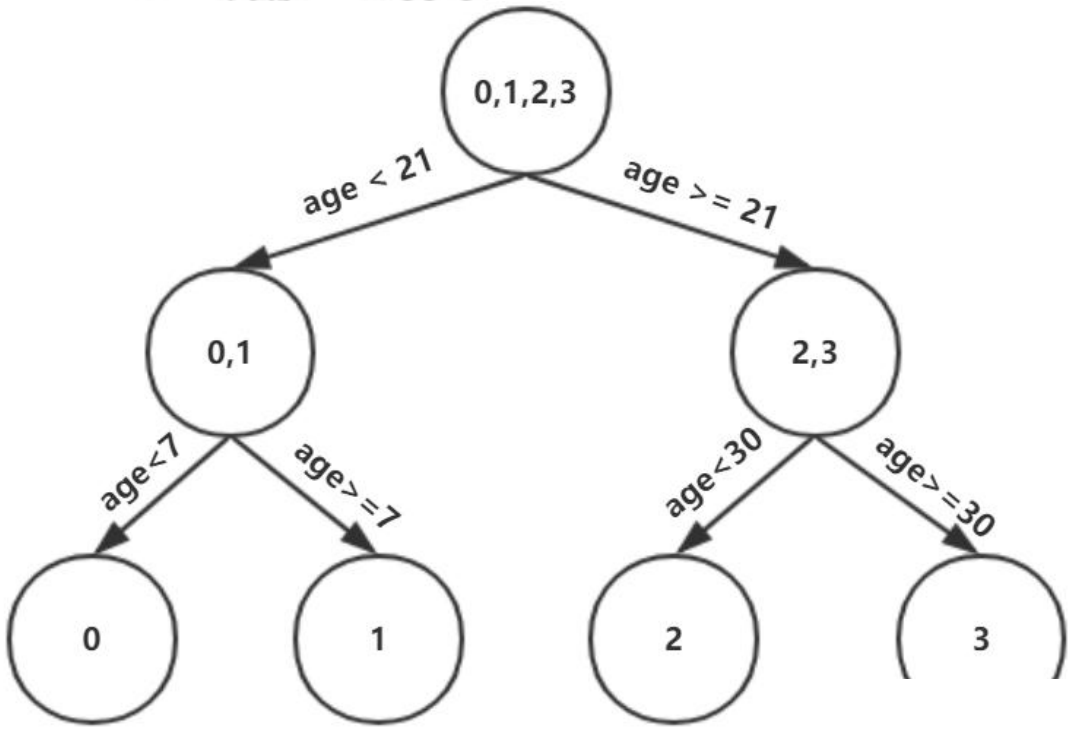


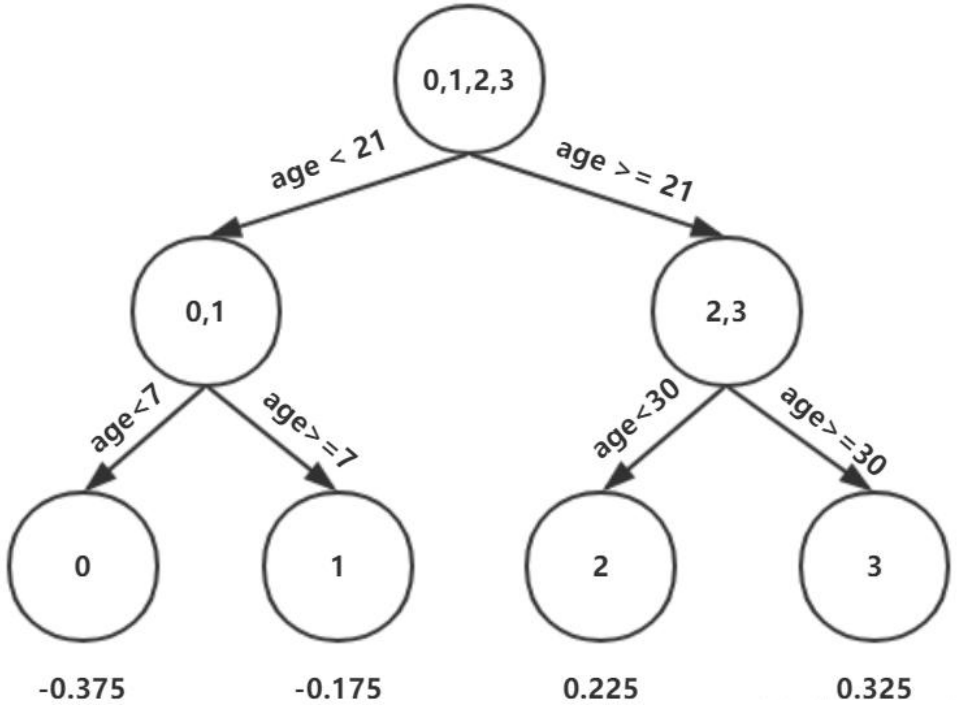
划分点	小于划分点的样本	大于等于划分点的样本	SE_l	SE_r	SE_{sum}
年龄21	/	2，3	0	0.005	**0.005**
年龄30	2	3	0	0	**0**
体重60	/	2，3	0	0.005	**0.005**
体重70	3	2	0	0	**0**

对于右节点，只含有2,3两个样本，根据表选择年龄30划分（也可以选体重70）

NeuHub AI Links All

梯度提升树（GBDT）





给这每个叶子节点分别赋值，来拟合残差。
平方损失求导，令导数等于零，化简之后得到每个叶子节点的值其实就是标签值的均值。

梯度提升树（GBDT）

- 更新强学习器，需要用到参数学习率：learning_rate=0.1，用lr表示。
- 如果每次都全部加上（学习率为1）很容易一步学到位导致过拟合。

$$f_1(x) = f_0(x) + lr * \sum_{j=1}^{4} \gamma_{j1} I(x \in R_{j1})$$

NeuHub
AI Links All

梯度提升树（GBDT）

重复上述步骤，直到m>5 结束，最后生成5棵树,得到最后的强学习器：

$$f_5(x) = f_0(x) + lr * \sum_{m=1}^{5} \sum_{j=1}^{4} \gamma_{jm} I(x \in R_{jm})$$

NeuHub AI Links All

扩展：GBDT与Xgboost的区别

- 基分类器的选择：传统GBDT以CART作为基分类器，XGBoost支持线性分类器。
- 二阶泰勒展开：传统GBDT在优化时只用到一阶导数信息，XGBoost则对代价函数进行了二阶泰勒展开。
- 方差-方差权衡：XGBoost在目标函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出分数的L2模的平方和，使学习出来的模型更加简单，防止过拟合。

回顾

- 集成学习：Bagging，Boosting
- 提升树是什么？
- 梯度提升树与提升树的差异？
- 扩展：Xgboost

京东智联云 & 贪心科技 GREEDY TECHNOLOGY
NeuHub AI Links All

参考

GBDT

https://www.douban.com/note/670632222/

https://blog.csdn.net/zpalyq110/article/details/79527653

https://www.cnblogs.com/pinard/p/6140514.html

XGboost

https://www.cnblogs.com/pinard/p/10979808.html

why 负梯度

https://www.zhihu.com/question/63560633

NeuHub AI Links All